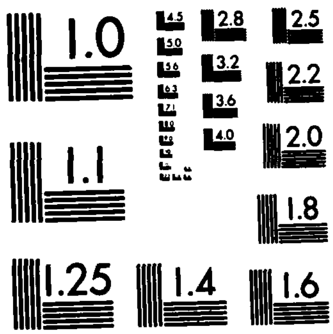
AD-A123 727 COMPUTATIONAL COMPARISON OF VALUE ITERATION ALGORITHMS FOR DISCOUNTED MARKOV DECISION PROCESSES(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA L C THOMAS ET AL. DEC 82 NPS55-82-034 1/1

UNCLASSIFIED F/G 12/1 NL

END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

NPS55-82-034

NAVAL POSTGRADUATE SCHOOL

Monterey, California

COMPUTATIONAL COMPARISON OF VALUE ITERATION ALGORITHMS FOR DISCOUNTED MARKOV DECISION PROCESSES

by

L. C. Thomas
R. Hartley
A. C. Lavercombe

December 1982

Approved for public release; distribution unlimited

Prepared for:

Naval Postgraduate School
Monterey, Ca 93940

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Reproduction of all or part of this report is authorized.

Lyn C. Thomas

L. C. Thomas
University of Manchester
Manchester, U.K.

R Hartley

R. Hartley
University of Manchester
Manchester, U.K.

A.C. Lavercombe

A. C. Lavercombe
Bristol Polytechnic
Bristol, U.K.

Reviewed by:

K. T. Marshall

K. T. Marshall, Chairman
Department of Operations Research

Released by:

William M. Tolles

William M. Tolles
Dean of Research

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NPS55-82-034	2. GOVT ACCESSION NO. A123 727	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) COMPUTATIONAL COMPARISON OF VALUE ITERATION ALGORITHMS FOR DISCOUNTED MARKOV DECISION PROCESSES		5. TYPE OF REPORT & PERIOD COVERED Technical 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) L. C. Thomas R. Hartley A. C. Lavercombe		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61152N; RR000-01-10 N0001483WR30104
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, Ca 93940		12. REPORT DATE December 1982 13. NUMBER OF PAGES 10
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computational comparison
Value iteration
Decision processes

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper describes a computational comparison of value iteration algorithms for discounted Markov decision processes.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

COMPUTATIONAL COMPARISON OF VALUE ITERATION ALGORITHMS

FOR DISCOUNTED MARKOV DECISION PROCESSES

L. C. Thomas*
R. Hartley

Department of Decision Theory, University of Manchester,
Manchester, U.K.

and

A. C. Lavercombe
Department of Mathematics, Bristol Polytechnic,
Bristol, U.K.

*At present: Department of Operations Research, Naval Postgraduate School, Monterey, CA 93940, USA. He is a National Research Council ,Research Associateship Fellow.

Abstract: This paper describes a computational comparison of value iteration algorithms for discounted Markov decision processes.

1. INTRODUCTION

This note describes the results of a computational comparison of value iteration algorithms suggested for solving finite state discounted Markov decision processes. Such a process visits a set of states $S = \{1,2,\ldots M\}$. When it is in state i, one can choose an action k from the finite action set K_i, and then receive an immediate reward r_i^k and with probability p_{ij}^k the process will be in state j at the next period. The object is to maximize, $v(i)$, the maximum discounted reward over an infinite horizon starting in state i, where β is the discount factor. It is well known [1] that $v(i)$ satisfies the optimality equation

$$v(i) = \max_{k \in K_i} \left\{ r_i^k + \beta \sum_{j=1}^{M} p_{ij}^k \, v(j) \right\} \qquad (1.1)$$

We record the time for value iteration algorithms to obtain ε-optimal solutions, v_n, to (1.1), (i.e. $|v_n - v|_\infty < \varepsilon$, where $|v|_\infty = \max_i |v(i)|$) on randomly generated problems. We look at three classes of fifteen problems each with $\beta = 9$ and $\varepsilon = .0001$, where $v(i) \gtrsim 2{,}000$. Class 1 problems have 100 states and between 2 and 7 actions per state; class 2 have 40 states and between 2 and 70 actions per state, whereas class 3 have 10 states and up to 500 actions per state. Details of how the problems are generated and computing facilities used are given in [12].

In Section two we describe the schemes examined and the various bounds that can be used for stopping them. Section three concentrates on one scheme that did well in the comparison - ordinary value iteration - and looks at various methods for eliminating non-optimal actions both permanently and temporarily.

2. SCHEMES AND BOUNDS

The scheme usually described as value iteration is

$$v_{n+1}(i) = \max_{k\epsilon K_i}\left\{v_n^k(i)\right\} = \max_{k\epsilon K_i}\left\{r_i^k + \beta \sum_{j=1}^{M} p_{ij}^k \; v_n(j)\right\} \tag{2.1}$$

which was discussed in [1,3]. In analogy with the notation of linear equations we call this Pre-Jacobi (PJ). This analogy leads us to think of the following alternative schemes.

Jacobi (J):

$$v_{n+1}(i) = \max_{k\epsilon K_i}\left\{(r_i^k + \beta \sum_{j\neq i} p_{ij}^k \; v_n(j))/(1-\beta p_{ii}^k)\right\} \tag{2.2}$$

Pre-Gauss-Seidel (PGS):

$$v_{n+1}(i) = \max_{k\epsilon K_i}\left\{r_i^k + \beta \sum_{j=1}^{i-1} p_{ij}^k \; v_{n+1}(j) + \beta \sum_{j=i}^{M} p_{ij}^k \; v_n(j)\right\} \tag{2.3}$$

Gauss-Seidel (GS):

$$v_{n+1}(i) = (A_{GS}v_n)(i) = \max_{k\epsilon K_i}\left\{(r_i^k + \beta \sum_{j=1}^{i-1} p_{ij}^k v_{n+1}(j) + \beta \sum_{j=i+1}^{M} p_{ij}^k \; v_n(j))/(1-\beta p_{ii}^k)\right\} \tag{2.4}$$

Successive over Relaxation (SOR):

$$v_{n+1}(i) = \omega(A_{GS}v_n)(i) + (1-\omega)v_n(i) \tag{2.5}$$

(PGS) was suggested by Kushner [4], Porteus [8] and Reetz [10]; (J) and (GS) is found in [9] and SOR in [5]. Experiments with SOR suggested a value of $\omega = 1.28$ for robust and speedy convergence.

We require bounds on the iterates of the scheme to ensure we stop when v_n is within a specified value of the optimal v of (1.1). One can use the L_∞ norm bound, which says if $||Q||_\infty \le \alpha < 1$ for all possible transition matrices in the scheme

$$\underline{v}_{n+1} = \max_k \{\underline{s}^k + Q^k \underline{v}_n\} \tag{2.6}$$

then $|v - v_{n+1}|_\infty \le \alpha \; |v_{n+1} - v_n|_\infty/(1-\alpha)$. For (PJ), (J), (PGS) and (GS), it is trivial to show the corresponding Q's have L_∞ norm less then β . For S.O.R. we estimate α by $|v_{n+1} - v_n|_\infty / |v_n - v_{n-1}|_\infty$ and substitute in (2.6) to get a heuristic bound.

Porteus [7] described tighter bounds for these schemes, exploiting the non-negativity of the elements q_{ij} of Q in (2.6). They require calculation of $\alpha_i^k = \sum_{j=1}^{M} q_{ij}^k$ for the maximizing action k at each iterate, and we call these the P.C. bounds - (Porteus with calculation). In [12] we describe how to estimate the α_i^k initially, which avoids the calculation at each step, but gives looser bounds, which we denote P.N.C. - (Porteus no calculation). For the (PJ) scheme we also use the second order bounds (S.O.) described in [11], which uses the last three iteration values to get a tighter lower bound than Porteus's bound.

The results are given in the following table where (Av) is the average C.P.U. time for solving the fifteen problems, S.D. the standard deviation of the C.P.U. time, and N the number of problems that method was quickest at solving.

TABLE 1

METHOD	BOUNDS	CLASS 1 (100 STATE)			CLASS 2 (40 STATE)			CLASS 3 (10 STATE)		
		AV.	S.D.	N.	AV.	S.D.	N.	AV.	S.D.	N.
PJ	PC=PNC	1.66	.11	15	.79	.10	14	.54	.07	4
	SO	1.69	.11	0	.80	.11	1	.54	.08	11
J	L_∞	11.88	.59	0	14.38	2.27	0	7.66	1.14	0
	PNC	10.49	.63	0	11.55	2.05	0	6.90	1.09	0
	L_∞	6.86	.33	0	8.62	1.34	0	5.18	.82	0
PGS	PC	6.59	.33	0	8.34	1.32	0	5.03	.81	0
	PNC	6.60	.33	0	8.40	1.33	0	5.01	.81	0
	L_∞	6.55	.32	0	8.13	1.41	0	3.99	.61	0
GS	PC	6.32	.34	0	7.77	1.38	0	3.90	.61	0
	PNC	6.25	.32	0	7.70	1.41	0	3.83	.61	0
SOR	L_∞	3.30	.22	0	4.00	.55	0	1.86	.30	0

For Jacobi, the P.C. bound is the same as the P.N.C. bound and so the latter must be faster as it involves less calculation. It is obvious from Table 1 that P.J. with Porteus bounds performs very well, and in the next section we concentrate on this scheme and apply elimination of non-optimal actions.

3. ACTION ELIMINATION

MacQueen [6] described how for any bounds one can observe a test to identify actions that cannot optimize the right hand side of (1.1) and so can be permanently eliminated from the calculation. Applying MacQueen's bounds [6] and Porteus's bound [7] for the PJ algorithm leads to the following tests to eliminate action k in K_i permanently.

MacQueen $\quad v_n^k(i) < v_n(i) + \beta(a_n - b_n)/(1-\beta)$ (3.1)

Porteus $\quad v_n^k(i) < v_n^d(i) + \beta^2(a_{n-1} - b_{n-1})/(1-\beta)$ (3.2)

where $a_n = \min_i(v_n(i) - v_{n-1}(i))$, $\quad b_n = \max_i(v_n(i) - v_{n-1}(i))$.

We looked at four ways of implementing these tests.

M1. At n^{th} iteration, calculate and store $v_n(i)$ for each i . Then calculate a_n and b_n . Recalculate $v_n^k(i)$ and use (3.1) to test for elimination.

M2. At n^{th} stage, calculate and store all $v_n^k(i)$. Hence calculate $v_n(i)$, a_n , b_n and test for elimination using (3.1) without recalculating $v_n^k(i)$.

P1. At $n+1^{th}$ stage, calculate $v_{n+1}^k(i)$, starting with action k that maximized $v_n^k(i)$ at previous stage. Apply (3.2) as soon as you calculate each $v_{n+1}^k(i)$ using as d the one that gives maximum $v_{n+1}^k(i)$ so far calculated, see [7].

P2. At $n+1^{th}$ stage, calculate and store $v_{n+1}^k(i)$. Then using $v_{n+1}^d(i) = v_{n+1}(i)$ apply (3.2).

As Table 2 shows M2 is far superior to M1, but P1 and P2 give similar results. All three cut the average time by a half though.

Hastings and Van Nunen [2] pointed out that one could also eliminate actions temporarily, i.e. actions that will not be the optimizing actions at the next iteration of the PJ algorithm. This is based on the inequality

$$v_{n+s}^k(i) - v_{n+s}^k(i) \geq v_n(i) - v_n^k(i) - \beta \sum_{j=1}^{s} (b_{n+j-1} - a_{n+j-1}) \tag{3.3}$$

If the R.H.S. of (3.3) is positive k will not optimize the $n + s^{\underline{th}}$ iteration, and in that case, at the $n + s + 1^{\underline{th}}$ iteration we need only subtract another $\beta(b_{n+s} - a_{n+s})$ from this positive number to test if k could be optimal. If action k is not eliminated at the $n + s^{\underline{th}}$ iteration, $v_{n+s}(i) - v^k_{n+s}(i)$ is stored for the test at the next iteration. We looked at four ways of implementing these two elimination procedures. Recall that the $n + 1^{\underline{th}}$ iteration consists of the following sequence of calculations.

$$a_n, b_n \xrightarrow{\text{(I)}} v^k_{n+1}(i) \xrightarrow{\text{(II)}} v_{n+1}(i) \xrightarrow{\text{(III)}} a_{n+1}, b_{n+1} \longrightarrow v^k_{n+2}(i)$$

TEMP HVN. Hastings and Van Nunen [2] suggested the temporary elimination test be made at (I) and if k was not temporarily eliminated then $v^k_{n+1}(i)$ was calculated. The permanent elimination test was made at (II) using (3.2) with $v^d_{n+1}(i)$ replaced by a lower bound $v_n(i) + \beta a_n$. If the action is not permanently eliminated, $v_n(i) + \beta a_n - v^k_{n+1}(i)$ (rather than $v_{n+1}(i) - v^k_{n+1}(i)$) is stored TEMP + P1. Temporary elimination occurs at (I) and permanent elimination at (II) using P1 .

TEMP + P2. Again this has temporary elimination at (I) and permanent elimination at (III) using P2 .

TEMP + M2. Temporary elimination occurs at (I) and in this case $v^k_{n+1}(i)$ was stored until (IV) and then the M2 technique used. If action k was not eliminated $v_{n+1}(i) - v^k_{n+1}(i)$ was stored for the temporary elimination test of the next iteration, which followed immediately.

In this case when permanent and temporary elimination are done at the same stage, it is obvious that any action which is permanently eliminated would also be temporarily eliminated.

This leads us to ask is it worth permanently eliminating, so

TEMP ONLY Perform the temporary elimination test at (I). If action k is not eliminated $v^k_{n+1}(i)$ is stored through state (II) and changed to $v_{n+1}(i) - v^k_{n+1}(i)$ at stage (III) for the next temporary elimination test at (IV).

Table 2 describes the results and shows that temporary elimination further cuts the time by 25%, and that pure temporary elimination might be particularly good on large scale problems.

TABLE 2

METHOD	CLASS 1 (100 STATE)			CLASS 2 (40 STATE)			CLASS 3 (10 STATE)		
	AV.	S.D.	N.	AV.	S.D.	N.	AV.	S.D.	N.
M1	1.51	.09	0	0.67	.08	0	0.42	.06	0
M2	0.81	.05	15	0.36	.04	15	0.25	.03	15
P1	0.87	.05	0	0.43	.05	0	0.26	.03	0
P2	0.88	.05	0	0.45	.05	0	0.28	.04	0
TEMP HVN	0.62	.03	0	0.27	0.03	0	0.21	.03	0
TEMP + P1	0.60	.04	0	0.25	.02	3	0.20	.03	11
TEMP + P2	0.58	.03	0	0.26	.02	0	0.23	.03	0
TEMP + M2	0.59	.04	0	0.22	.04	2	0.21	.03	4
TEMP ONLY	0.55	.03	15	0.22	.04	10	0.21	.03	0

Our object has not been to obtain a best buy, but to give some idea of the merits of the various schemes, bounds and improvements. Obviously for more structured problems, algorithms which exploit the structure will be at an advantage.

ACKNOWLEDGEMENTS

The authors are grateful to the Social Science Research Council for their financial support. We are also indebted to Professor D. J. White and Dr. S. French for stimulating discussions.

REFERENCES

1. BLACKWELL, D., "Discounted Dynamic Programming", Ann. Math. Stat. 36, 226-235, (1965).

2. HASTINGS, N. A. J., VAN NUNEN, J. A. E. E., "The action elimination algorithm for Markov decision processes: Markov Decision Processes" ed. by H. C. Tijms, J. Wessels, Mathematical Centre Tract No. 93, Amsterdam, pp 161-170, (1977).

3. HOWARD, R., Dynamic Programming and Markov Processes, Wiley, New York, (1960).

4. KUSHNER, H., Introduction to Stochastic Control, Holt, Rinehart and Winston, New York, (1971).

5. KUSHNER, H., Kleinman, A. J., "Accelerated Procedures for the solution of discrete Markov control problems", I.E.E.E. Trans. on Automatic Control 16, 147-152, (1971).

6. MACQUEEN, J., "A test for sub-optimal actions in Markovian Decision Problems", Opns. Res. 15, 559-561, (1967).

7. PORTEUS, E., "Some bounds for sequential decision processes'", Man. Sci. 18, 7-11, (1971).

8. PORTEUS, E., "Bounds and transformations for finite Markov decision chains", Opns. Res. 23, 761-784, (1975).

9. PORTEUS, E., TOTTEN, J., "Accelerated computation of the expected discounted return in a Markov chain", Opns. Res. 26, 350-358, (1978).

10. REETZ, D., "Approximate solutions of a discounted Markovian decision process", Dynamische Optimierung, Bonner Math. Schoiften, 98, pp. 77-92, (1977).

11. THOMAS, L. C., "Second order bounds for Markov decision processes", J. Math. Anal. Appl. 80, 294-297, (1981).

12. THOMAS, L. C., HARTLEY, R., LAVERCOMBE, A., "Computational Comparison for discounted Markov decision processes-value iteration. Notes in Decision Theory, No. 100, Dept. of Decision Theory, Univ. of Manchester, Manchester, (1982).

DISTRIBUTION LIST

	NO. OF COPIES
Library, Code 0142 Naval Postgraduate School Monterey, CA 93940	4
Dean of Research Code 012A Naval Postgraduate School Monterey, CA 93940	1
Library, Code 55 Naval Postgraduate School Monterey, CA 93940	2
Professor L. C. Thomas Code 55 Naval Postgraduate School Monterey, CA 93940	60